한국 CAD/CAM 학회 논문집
제 12권 제 6호 2007년 12월 pp. 413-421

학술논문

모바일 장치에서의 가시화를 위한 경계기반 삼각화

양상욱*, 최 영**

A Constrained Triangulation Technique for Visualization on Mobile Devices

Sangwook Yang* and Young Choi**

ABSTRACT

3D rendering is becoming a common feature of mobile application programs with the rapid advance of mobile devices. Since the existing rendering engines do not provide triangulation functions, mobile 3D programs have focused on an efficient handling with pre-tessellated geometry. In addition, triangulation is comparatively expensive in computation, so it seems that the triangulation cannot be easily implemented on mobile devices with limited resources. Triangulation of 3D geometry is the essential process of visualization of 3D model data and many different triangulation methods have been reported. We developed a light and fast visualization process that involves constrained triangulation based on Voronoi diagram and applied it to a mobile computer application. In this paper, we applied *kd*-tree to the original incremental construction algorithm and produced new **O**(*n*log*n*) incremental construction algorithm. And we show a simple and efficient constrained triangulation method based on Voronoi diagram. This paper also describes an implementation of mobile STEP data viewer as an application of our proposed algorithms.

Key words : Visualization, Mobile device, Constrained triangulation

1. 서 론

PDA나 스마트폰 등의 모바일 장치들의 보급이 증가하고 모바일 장치의 하드웨어 및 소프트웨어 기술이 발전함에 따라 모바일 3D기술도 PC 수준에 가까이 접근하고 있지만, 모바일 장치는 하드웨어와 소프트웨어 환경이 PC나 워크스테이션 보다 제한적이라는 점에서 데스크탑 환경과는 다른 방식으로 접근할 필요가 있다. 모바일 3D 구현의 초기에는 간소화된 3D 형상 데이터를 이용하거나[1], 모델링이나 삼각화 또는 렌더링 등의 많은 계산이 필요한 것은 원격 서버에서 수행하고 모바일 장치에서는 이미지를 보여주는 방식[2]의 연구가 진행되었다. 2004년과 2005년 사이에 OpenGL의 임베디드 시스템 버전인 OpenGL|ES[3]의 규격과 공개된 API가 제공되고, 마이크로 소프트의 Windows Mobile 5.0과 더불어 DirectX Mobile[4] API도 출시됨에 따라 모바일 장치에서 삼각화된 3D 형상의 직접 렌더링이 편리해졌다. 이들 렌더링 라이브러리들을 이용하여 CAD데이터를 가시화하기 위해서는 B-rep 솔리드나 드림 곡면과 같이 경계 기반의 곡면 형상으로부터 삼각화 데이터를 얻어내는 과정을 거쳐야 한다.

이 삼각화 데이터는 가시화뿐만 아니라 CAE 환경에서도 사용되며, 1970년대 이후로 현재까지 많은 연구가 진행되어 온 만큼 많은 소프트웨어들이 상용화되거나 공개 소프트웨어로 구현되어 있다[5].

3D모델의 삼각화에 대한 연구는 CAE 분야에서 해석을 위한 관점에서 주로 이루어져 왔으며, 해석의 정밀도를 높이기 위한 정규화된(regular) 삼각형을 잘 만들어 내는데 주안점을 둔 연구가 많이 수행되었다[6-8].

또한 모바일 장치에서의 3D 구현의 예를 보면 STL과 같은 정적 삼각형 데이터를 가시화 하거나 프로그레시브 메시를 이용하여 동적인 LOD를 적용하여 가시화하는 사례[9]를 찾아볼 수 있는데, 이들은 이미 삼각화된 데이터에서부터 출발하고 있으며 모바일 장치에서 직접 삼각화를 구현한 사례는 현재까지 없다.

*교신저자, 학생회원, 중앙대학교 기계공학부
**종신회원, 중앙대학교 기계공학부
- 논문투고일: 2007. 04. 11
- 심사완료일: 2007. 08. 20

CAD 분야에서 가시화를 위한 삼각화는, CAE 분야에서 사용되는 정규화된 삼각형을 얻기 보다는 경계 곡선과 곡면의 곡률을 제대로 표현하면서, 전체적인 삼각형의 수가 적도록 하는 것이 프로그램의 성능 및 효율 측면과 네트웍 상에서 가시화 데이터의 공유의 측면에서 유리하다고 할 수 있다. 일반적으로 가시화와 CAE를 위한 3D 형상의 삼각화는 3D 모델링 좌표계로부터 2D 매개변수 공간으로 매핑한 후 2D 삼각화 알고리즘을 적용하여 다시 3D 공간으로 되돌리는 방법이 많이 사용되고 있지만[6-8], 3D 곡면상에서 직접 메시 생성을 수행하는 방법들도 있다[10,11].

본 논문에서는 매개변수 공간상에서의 삼각화를 사용하고 있는데, Fig. 1에는 본 논문에서 적용한 CAD 데이터의 가시화 단계를 도식화하고 있다.

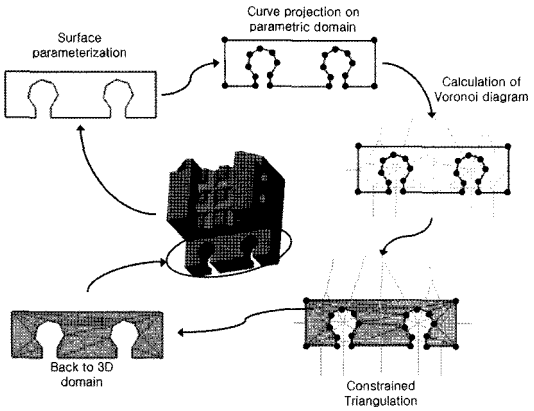

Fig. 1. Visualization process.

Fig. 1에는 B-rep 솔리드 표현의 CAD 데이터로부터 곡면과 경계 형상을 2D 매개변수화된 도메인으로 옮긴 후 보로노이 다이어그램을 계산하고 이를 기반으로 경계 형상을 표현할 수 있도록 삼각화하는 과정이 나타나 있다. 본 논문에서는 이러한 전 과정이 모바일 장치에서 수행될 수 있도록 구현하는 데에 적용 가능한, 간결한 보로노이 다이어그램 계산 알고리즘과 가시화 데이터를 위한 경계기반 삼각화 방법을 설명한다. 또한 가시화 전 과정을 PDA 장치에서 수행될 수 있도록 구현한 예를 제시한다.

2. 보로노이 다이어그램과 삼각화

보로노이 다이어그램을 구하는 방법 중 가장 효율적이라고 알려진 방법은 최대 $O(n\log n)$ 이하의 시간 복잡도를 보장하는 Fortune의 방법[12]이다. Fortune의 방법은 모든 점(site)이 정해졌을 때 현재까지 알려진 가장 효율적인 방법이지만, 곡면의 경계를 결정하는 모든 샘플링 포인트를 가지고 보로노이 다이어그램을 결정한 후에 경계 에지에 의한 구속 조건을 별도로 계산하는 것 보다는 보로노이 다이어그램을 만드는 단계에서부터 경계 조건을 고려하여 전체 프로세스를 단축시키는 것이 효율적이기 때문에 본 논문에서는 점진적 구성(incremental construction)[13,14] 방법을 향상시켜서 사용한다.

2.1 보로노이 다이어그램

$P=\{p_1, p_2, \cdots, p_n\}$를 2차원 유클리디언 평면에서의 점($p_i$, site)들의 집합으로 정의하면, 평면상의 모든 점은 그 점에서 가장 가까운 사이트가 어느 것인지 결정할 수 있게 되며 이를 식으로 나타내면 식 (1)과 같다.

$$V(p_i) = \{x; |p_i - x| \le |p_j - x| \forall j \neq i\} \quad (1)$$

$V(p_i)$가 식 (1)을 만족한다면 $V(p_i)$를 보로노이 영역(Voronoi region)이라고 부르며, p_i가 가장 가까운 사이트가 되는 점들의 집합이다. 또한 $V(p_i)$의 집합 $V(P)$는 보로노이 다이어그램이 된다.

보로노이 다이어그램이 가진 여러 특성 들 중 본 연구와 관련된 중요한 특성 몇 가지만 인용하자면 다음과 같다[15].

V1. v가 $V(p_1)$, $V(p_2)$, $V(p_3)$의 교점이면, v는 p_1, p_2, p_3를 지나는 원 $C(v)$의 중심이 된다.
V2. $C(v)$는 Delaunay 삼각형의 외접원이 된다.
V3. $C(v)$ 내부에는 사이트가 존재하지 않는다.

2.2 Delaunay 삼각화

Delaunay는 $C(P)$에 대한 직선들을 그리면 보로노이 사이트 집합 P에 대한 삼각화가 됨을 보였다. 이 삼각형 집합을 Delaunay 삼각화(triangulation) $D(P)$라 하면, $D(P)$와 $V(P)$의 관계는 상동(dual)이 된다. Delaunay 삼각화의 특성 중 본 연구에서 중요하게 사용된 것들은 다음과 같다[15].

D1. $D(P)$는 $V(P)$의 상동이다. (정의에 의해)
D2. $D(P)$의 삼각형은 $V(P)$의 버텍스에 대응한다.
D3. $D(P)$의 에지는 $V(P)$의 에지에 대응한다.
D4. $D(P)$의 노드는 $V(P)$의 각 영역에 대응한다.

2.3 점진적 구성 알고리즘

점진적 구성은 원래의 보로노이 다이어그램 V가 k

개의 사이트를 가지고 있다고 할 때, 새 사이트 p가 추가된 후의 보로노이 다이어그램 V'를 찾는 문제이다. p가 원들 $C(v_1)$, …, $C(v_m)$안에 떨어졌다면, V의 보로노이 버텍스 v_1, …, v_m 은 V'의 보로노이 버텍스가 될 수 없으며, 이들을 포함하는 보로노이 에지 또한 V'의 보로노이 에지가 될 수 없으므로 $C(v_1)$, …, $C(v_m)$을 이루는 사이트들과 p에 의한 보로노이 영역이 재계산되어야 한다. Fig. 2에는 점진적 구성의 예가 나타나 있다. 새로운 점이 추가되었을 때, 그 점을 포함하는 보로노이 원을 검색하여 두 개의 원을 찾고, 그 원들에 관계되는 원래의 보로노이 영역을 다시 계산하게 된다.

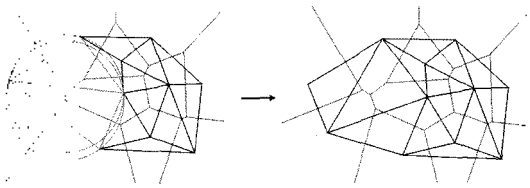

Fig. 2. Incremental construction.

점진적 구성에서 하나의 사이트가 추가되는 시간 복잡도는 새 사이트가 포함되는 원들을 찾기 위해서 $\mathbf{O}(n)$이 소요되므로 전체의 시간 복잡도는 $\mathbf{O}(n^2)$이 된다.

3. 가시화를 위한 경계기반 삼각화

고전적인 점진적 구성 방법은 $\mathbf{O}(n^2)$의 시간 복잡도를 갖지만 현재의 일반적인 점진적 구성 방법은 검색 트리 등을 이용하여 최대 $\mathbf{O}(n\log n)$의 시간 복잡도를 가지며 실행시간 $\mathbf{O}(n)$ 시간 복잡도를 보일 수 있다. 본 논문이 제시하는 알고리즘 또한 동일한 시간 복잡도를 가지는데, 새 점이 들어갈 보로노이 영역을 찾는 과정에서 *kd*-트리를 적용하여 검색 트리를 위한 메모리를 적게 사용하면서도 단순한 구현이 가능하도록 하였다.

보로노이 다이어그램으로부터 삼각화 데이터를 얻는 과정은 곡면 경계 곡선의 안쪽 삼각형에 대해서 보로노이 다이어그램과 Delaunay 삼각화의 상동 관계로 구하게 되며, 이 과정은 최대 $\mathbf{O}(n)$ 이하의 시간 복잡도를 갖는다.

3.1 데이터 구조

본 알고리즘은 보로노이 에지 구현을 위하여 하프에지(half-edge) 구조를 사용한다. 각 보로노이 영역은 *kd*-트리에 저장된 사이트를 참조하고, 하프에지의 리스트를 가지고 있다. 각 하프에지의 시작과 끝이 되는 보로노이 버텍스들은 글로벌하게 관리되며, 하프에지들은 자신의 메이트가 되는 하프에지의 참조를 가지고 있다. 이러한 구조를 통해서 한 사이트가 찾아졌을 때 그 사이트와 관련된 보로노이 영역 및 하프에지들과 보로노이 버텍스들, 그리고 인접 영역 등 모든 정보를 검색이 아닌 데이터 구조에 의해서 얻을 수 있다. Fig. 3에는 보로노이 영역과 하프에지, 버텍스들 간의 관계가 나타나 있다.

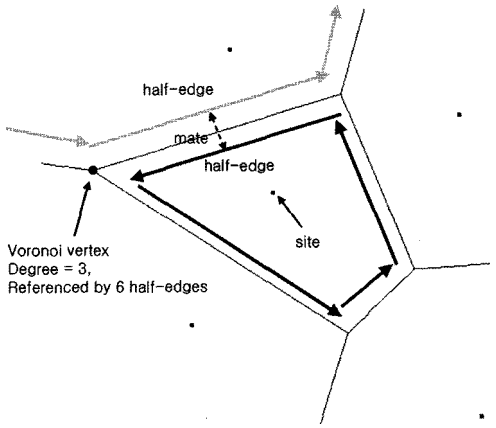

Fig. 3. Graphical description of the data structure for Voronoi diagram.

3.2 *kd*-트리를 적용한 점진적 구성 알고리즘

kd-트리는 *k*-dimensional 트리의 약어로, *k*-차원의 공간에서 공간 분할을 이용해서 좌표점들을 정렬하는 방법으로 BSP(Binary Space Partitioning)트리의 일종이다[16]. Bently는 *kd*-트리를 처음 소개한 그의 논문에서 구성된 트리로부터 최단거리 점을 찾아내는데 최대 $\mathbf{O}(\log n)$의 시간 복잡도를 가진다는 것을 보였다[17].

이러한 *kd*-트리의 특성과 식 (1)에 나타난 보로노이 다이어그램의 정의로부터, $V(P)$의 각 사이트가 *kd*-트리에 들어있다면, 새 점 p를 추가하기 위하여 p를 포함하는 기존의 영역 $V(p_i)$를 찾아 내는 것은, *kd*-트리에서 점 p에 대한 최단거리 점 p_i 를 찾는 것과 같으므로 $\mathbf{O}(\log n)$의 시간이 소요된다. 찾아낸 $V(p_i)$는 2.1절에서 언급한 보로노이 다이어그램의 특성 V3에 의해서 V'의 보로노이 영역이 될 수 없으므로 다시 계산되어야 하는데, 이 과정에서 추가적인 검색이 없이 인접 관계로부터 계산될 수 있다.

Field의 점진적 구성 알고리즘[14]은 새 사이트 p가 추가된 후 영향을 받는 $C(v_1)$, …, $C(v_m)$ 을 찾는데

$O(n)$을 소요하게 되고, 본 알고리즘은 새 사이트 p가 추가된 후 영향을 받는 $V(p_i)$를 찾는데 있어서 kd-트리를 이용한 최단거리 점 검색에 평균$O(\log n)$ 을 소요하게 된다.

Fig. 4에는 두 개의 영역을 가진 보로노이 다이어그램에 새로운 사이트가 추가되었을 때 세 개의 영역으로 나누어지는 간단한 예가 나타나 있다.

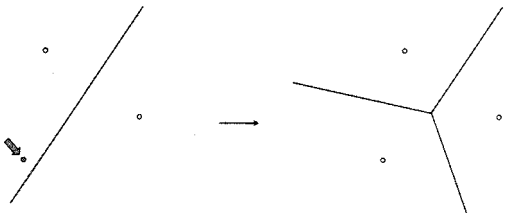

Fig. 4. Insertion example.

Fig. 5는 Fig. 4의 경우와 같이 새로운 사이트가 추가되었을 때 영향을 받는 기존의 보로노이 영역을 수정하고, 새 사이트에 의한 영역을 계산하는 세부 과정을 나타내고 있다. 각 과정에 대한 설명은 다음과 같다.

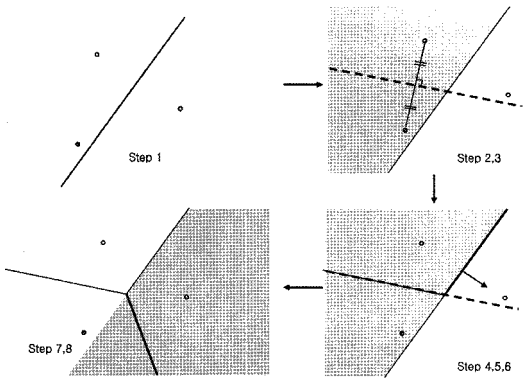

Fig. 5. Incremental process.

① 새 사이트가 추가되면 새 사이트가 포함되는 영역을 찾는다. 영역이 없다면 새 사이트를 중심으로 하는 유일한 영역이 생성되며 과정 종료된다. 찾은 영역은 수정될 영역 큐에 추가된다.

② 수정될 영역 큐로부터 하나의 대상 영역을 꺼낸다. 새 사이트와 대상 영역이 참조하는 사이트 사이의 하프플레인을 계산한다.

③ 대상 영역의 기존 하프에지 중 2단계에서 계산된 하프플레인과 교차하는 점이 있다면 교차점들을 찾는다. 교차점이 없으면 새 하프플레인이 수정할 영역의 하프 에지가 되며 7단계로 넘어간다.

④ 기존의 하프에지가 3단계에서 계산된 교차점에서부터 시작되도록 수정된다.

⑤ 4 단계에서 기존 하프에지가 메이트를 가지고 있다면, 그 메이트가 소속된 영역을 수정될 영역 큐에 추가한다.

⑥ 계산된 하프플레인과 교차점으로부터 새 하프에지를 생성한다.

⑦ 새 사이트가 소속된 새 영역에 7단계의 하프에지의메이트를 생성한다.

⑧ 수정될 영역 큐에 대기하는 영역이 있다면 2단계로 돌아가 반복한다.

2-8단계에 반복이 있는데, 이는 검색이 아닌 인접 정보로부터 직접 구하기 때문에 실행시간 계산에 포함되지 않는다. 전체 단계의 수행시간을 살펴보면 1단계에서 $O(\log n)$이고 2-8단계는 상수 복잡도이므로, 전체적으로 $O(\log n)$이 된다. 따라서 n개의 사이트가 추가되는 전체 시간 복잡도는 $O(n\log n)$이다.

3단계에서 교차점을 계산하는 경우 계산된 교차점과 일치하는 기존 교차점이 존재 한다면 이러한 교차점은 3개 이상의 보로노이 에지가 만나는 보로노이 버텍스가 되며, 이 경우에는 Delaunay 삼각형과 상동관계가 만들어 지지 않는다. 이러한 경우를 피하기 위해서 교차점을 계산하는 단계에서 Fig. 6에서와 같이 높은 차수가 되는 보로노이 점에 대해서 길이가 0인 가상의 에지를 삽입하여 삼각화 가능하도록 교차점의 중복 회수를 3으로 유지한다.

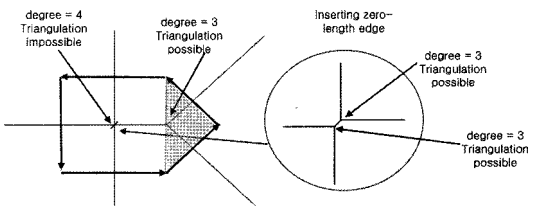

Fig. 6. Insertion of zero-length Voronoi edge.

3.3 경계기반 삼각화

보로노이 다이어그램과 Delaunay 삼각화의 정의에서 만들어진 보로노이 다이어그램으로부터 삼각형 데이터를 얻어내는 과정은 단순히 모든 에지들을 순회하면서 얻어낼 수 있다. Fig. 7에서는 내·외부 두 개의 경계 폴리곤에 의해 생성된 보로노이 다이어그램과 해당 Delaunay 삼각형을 보여주고 있다. 경계 폴

리곤의 세그먼트는 대부분 삼각형의 에지와 일치하는데, 왼쪽 그림에서 표시된 것과 같이 삼각형의 에지와 일치하지 않는 경계 세그먼트가 나타날 경우가 있다. 이때, 오른쪽 그림처럼 경계 세그먼트와 삼각형의 에지가 교차하는 부분에 새 사이트를 삽입하여 경계 에지와 삼각형의 에지가 일치하도록 만들어준다.

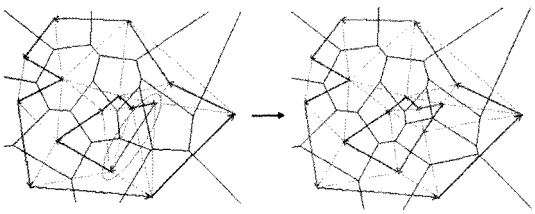

Fig. 7. Splitting constraint edge.

경계 폴리곤과 삼각형을 일치시킨 이후에, 경계 내부의 삼각형들만을 골라 내어서 삼각화를 완성하는 알고리즘은 다음과 같다.

① 모든 경계 에지를 큐에 넣는다.
② 큐에서 하나의 에지를 꺼내어, 이 에지를 밑변으로 하는 삼각형을 내부 삼각형으로 결정한다.
③ 내부 삼각형의 밑변을 제외한 두 변에 대해 경계 에지가 아닌 경우에 각 변의 메이트에 해당하는 에지를 경계 에지 큐에 삽입한다.
④ 큐의 사이즈가 3이상인 경우 2단계부터 반복한다.

Fig. 8의 상단 우측 그림은 알고리즘의 3단계를 보여 주고 있다. 이렇게 경계 내부의 삼각형을 선택하고, 선택된 삼각형의 에지 중 경계에 있지 않은 에지의 메이트에 대한 경계 에지를 만들어 주면서 경계 에지가 없어질 때까지 반복하면 그림의 화살표 진행 방향과 같이 내부 삼각형들을 모두 선별해 낼 수 있다.

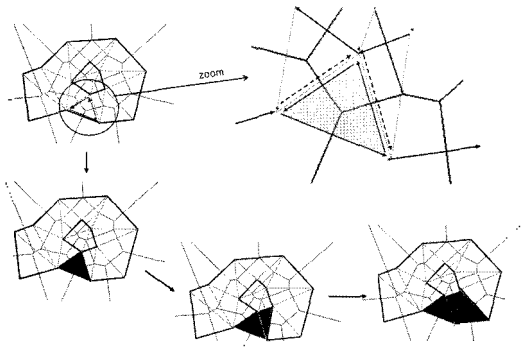

Fig. 8. Selecting internal triangles.

모바일 장치에서 실제 구현에서 가시화될 삼각형들을 선별해낸 이후에 저장 공간을 절약하기 위해서 해당 곡면의 보로노이 다이어그램은 삭제된다.

3.4 곡면 특성을 고려한 삼각화

모바일 장치와 같은 비교적 저성능의 컴퓨팅 환경에서 원활한 가시화 소프트웨어의 수행을 위해서 삼각화 결과의 삼각형의 개수가 적은 것이 좋다. 그러한 측면에서 3.3절에서 설명한 경계 기반의 삼각화는 곡면 경계 상의 점들로만 수행되는데 이 과정에서 곡면의 곡률(curvature)을 고려하지 않으면 다음의 Fig. 9와 같이 곡면을 제대로 표현하지 못하는 경우가 발생한다.

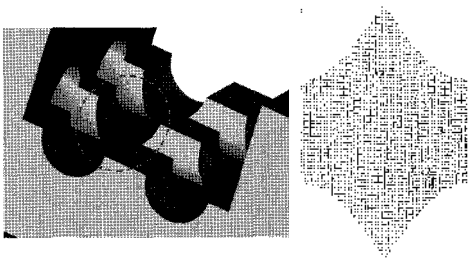

Fig. 9. Improper triangulation due to curvature on a cylindrical surface.

왼쪽 그림에서 원으로 표시된 부분은 실린더의 일부이며, 오른쪽 그림은 이 곡면의 경계와 삼각화 결과를 매개변수 좌표계에서 나타낸 것이다. 실린더의 경우 가로 방향 2차, 세로 방향 1차식으로 표현된다. 오른쪽 그림의 경계 곡선을 따라 살펴보면 곡선 요소에서 비교적 균등하게 샘플링된 점들을 따라 삼각화를 수행하고 있지만 곡면의 내부 곡률이 고려되지 않은 채 큰 삼각형들과 가로로 긴 삼각형들이 생성되어 그 결과로 왼쪽 그림에서 원통의 일부임을 제대로 표현하지 못하고 있다.

곡률 분제를 해결하는 일반적인 방법은 곡면의 곡률분포에 따라서 곡면 내부에 삼각형을 더 생성하는 적응적 메시 방법이다[5].

그러나 모든 곡면에 적응적 메시를 수행할 경우 삼각형의 수가 증가하는 단점이 있으므로 본 논문에서는 한 쪽 방향만 1차로 표현되는 곡면에 있어서 1차 방향으로는 부가적인 삼각형을 생성하지 않도록 하는 Fig. 10과 같은 방법을 고안하였다. 이렇게 한쪽 방향만 1차식으로 표현되는 곡면은 기계 부품에서 실린더나 콘, 스윕 곡면 등의 형태로 자주 나타난다.

Fig. 10의 왼쪽 그림과 오른쪽 그림은 같은 샘플링 포인트에 의한 경계로 표현되어 있다. 왼쪽 그림은 전체 실린더의 u, v 방향 매개변수의 범위를 모두 [0,1]로 표현하여 삼각화를 한 결과이며 Fig. 9와 같이 u 방향의 삼각화가 제대로 이루어지지 않고 있다. 오른쪽 그림은 2차인 u 방향은 [0,10]으로, 1차인 v 방향은 [0,1]로 매개변수화 하여 삼각화를 적용한 결과이다. 두 경우 삼각형의 수는 동일하지만 우측의 삼각화의 결과는 실린더의 곡률을 잘 표현하고 있음을 볼 수 있다.

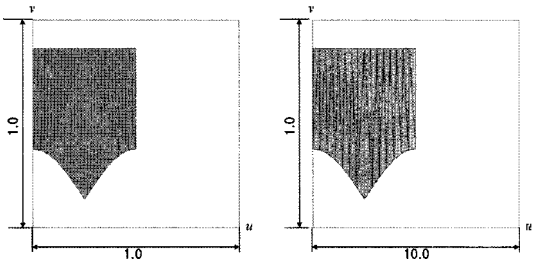

Fig. 10. Different triangulation results from parameter range variation.

4. 모바일 장치용 STEP 뷰어 구현

4.1 모바일 뷰어 구현

제 3장에서 기술된 삼각화 알고리즘을 이용하여 PDA(Personal Digital Assistant) 상에서 작동하는 STEP 뷰어를 구현한 구조는 Fig. 11과 같다.

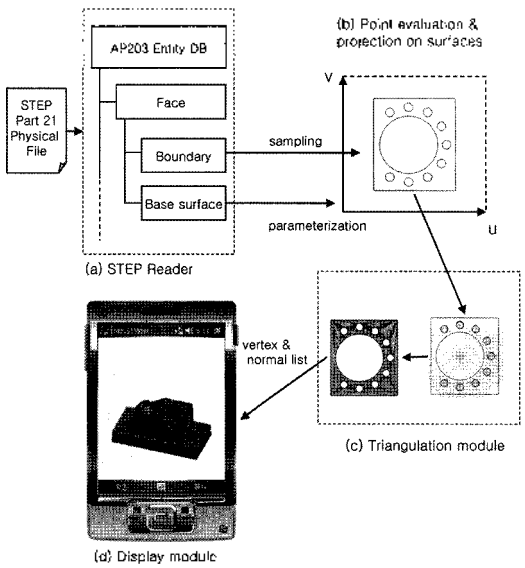

Fig. 11. The mobile STEP viewer.

Fig. 11(a)는 STEP 물리 파일로부터 B-rep 솔리드 구조의 데이터를 읽어 들여서 파싱한 후 AP203 엔티티의 객체 인스턴스로 변환하는 모듈이다. 이 과정에서 필요한 STEP 파일의 읽기와 엔티티 데이터베이스를 위한 PDA용 라이브러리는 공개된 바 없으므로, 직접 구현한 STEP 라이브러리를 사용하였다. 이STEP 라이브러리를 구현하기 위해서 STEP Tools 사의 ST-Developer처럼 EXPRESS 스키마로부터 C++ 엔티티 클래스를 생성하는 프로그램을 작성하였고, 이 프로그램에 의해서 AP203스키마로부터 변환된 C++ 클래스를 이용하여 STEP 파일을 읽고 객체 데이터로 다루는 라이브러리를 작성하였다. 자체 제작된 스키마 변환 프로그램과 STEP 파싱 라이브러리는 C++ 표준 함수만을 사용하여 PDA 환경으로 쉽게 이식 가능하도록 하였다.

Fig. 11(b)는 읽어온 경계 기반의 곡면데이터로부터 삼각화를 수행할 수 있도록 3차원 곡면들을 2차원 매개변수 형태로 표현하고 그 경계 곡선들을 샘플링하여 매개 변수 영역으로 사상(projection) 한다.

Fig. 11(c)에 나타나 있는 삼각화 모듈은 본 논문의 3장에 기술된 삼각화 방법을 적용하여 구현하였다. Fig 11(d)에서는 동적 3D 가시화를 위해 Hybrid Graphics 사의 OpenGL|ES 1.1 구현[10]을 이용하며 선택적으로 DirectX Mobile을 이용한 렌더링이 가능하도록 구현하였다. DirectX Mobile은 Windows Mobile 5.0 (WM5)에 그 런타임 라이브러리가 포함되어 있으므로 별도의 라이브러리를 모바일 장치에 배포하지 않아도 실행할 수 있는 장점이 있지만 WM5가 탑재된 PDA나 스마트폰 등의 기종에서만 사용 가능하다.

4.2 구현 결과

Fig. 12는 구현된 모바일용 STEP 뷰어가 실제 PDA 상에서 실행되고 있는 사진이다. 첫 번째 그림은 STEP 파일을 읽기 위한 대화상자 화면이고, 두 번째와 세 번째는 STEP 파일을 가시화한 결과이다. 개발 환경은 Pentium 4 Core 2 Duo 데스크탑 PC 상에서

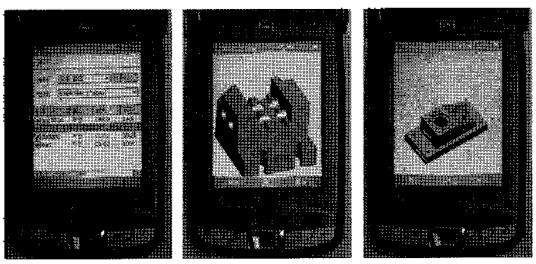

Fig. 12. Mobile STEP viewer on a PDA.

Microsoft Visual Studio 8.0을 이용하였으며, Windows Mobile 5.0 Pocket PC SDK와 OpenGL|ES 1.1 및 DirectX Mobile을 이용하여 구현하였다. 대부분의 자료 구조는 C++의 STL(Standard Template Library)[16]을 이용하였고, kd-트리는 범용성과 실행속도를 고려해 C++ 템플릿으로 작성하였다. 실행 및 테스트에 사용된 PDA는 HP 사의 hx2790b Pocket PC이며 한글 WM5가 탑재되어 있다.

4.3 수행 속도 분석

구현된 PDA용 STEP 뷰어의 성능 검사를 위해서 hx2790b 상에서 실행시간을 측정한 결과는 Table 1과 같다. 데이터 A와 B는 각각 Fig. 12의 중앙과 오른쪽에 나타나 있으며 시간의 단위는 모두 밀리세컨드(ms)이다. 렌더링 성능에 있어서 fps(frames per second)는 각 100회 그리기에 대한 평균 결과 값이다.

Table 1. Computation results on a PDA

		A	B
데이터	사이즈	68 kB	141 kB
	엔티티 수	1401	2755
	STEP 파싱 시간	2046	2704
형상변환	경계곡선 수	130	207
	곡선변환 시간	76	175
	곡면 수	44	99
	곡면변환 시간	130	338
삼각화	삼각화 시간	6543	30084
	결과삼각형 수	1060	3553
렌더링	OpenGL|ES	11.3 fps	7.2 fps
	DirectX Mobile	25.3 fps	16.3 fps

표의 결과를 항목 별로 살펴 보면, STEP 데이터를 읽는 시간은 두 데이터 사이에 차이가 크게 나지 않는데 A, B 두 데이터의 크기의 차이보다 저장소 입출력 속도에 더 영향을 받는 것으로 보인다. 형상 변환에 있어서는 전체 시간에 비해 적은 시간을 소요하고 있는데, 이는 테스트 대상 데이터에는 자유곡면/곡선을 포함하고 있지 않기 때문이며 추후 자유곡면을 포함한 데이터에 대해 수행한다면 좀 더 많은 시간이 소요될 것이다. 삼각화 시간은 2D 매개변수 공간상에서의 삼각화 시간뿐 아니라 삼각화된 점들에 대해3D공간상의 좌표와 법선 벡터를 구하는 시간을 포함하고 있는 값이다. 렌더링은 동일한 삼각형 데이터에 대해서 OpenGL|ES와 DirectX Mobile로 각각 렌더링한 결과를 나타내는데 DirectX Mobile이 약 2배 이상 빠른 것으로 나타난다. 이러한 결과로 보아서 3D 그래픽 엔진을 선택할 때, 여러 시스템에서 범용으로 사용하기 위해서는 OpenGL|ES을 사용하고 WM5가 탑재된 모바일 시스템에는 DirectX Mobile을 사용하는 것이 유리하다고 할 수 있다.

Fig. 13에는 매개변수 공간상에서의 삼각화 시간만을 좀 더 구체적으로 나타내었다. A와 B의 두 모델을 가시화 하는 과정에서 처리한 곡면들을 대상으로 곡면 경계의 복잡성과 삼각화의 실제 수행 시간 간의 상관 관계를 나타낸 것이다. 가로축은 곡면 경계 곡선을 샘플링하여 나눈 세그먼트의 수를 나타내고 세로 축은 해당 곡면을 삼각화하는 시간을 ms 단위로 나타내고 있다. 60세그먼트 근방에서 비슷한 입력에 대한 수행 속도의 차이가 크게 분포되어 있다. 이는 보로노이 다이어그램 구성 이후 3.3절의 Fig. 7에서 설명된 불일치점들을 찾아내는 과정에서 그 점들의 발생 빈도에 따른 차이이며 복잡한 형상의 경계를 가진 곡면에서 그 빈도가 높다. 추후 보로노이 다이어그램을 구성한 후 경계 에지를 검색하는 과정을 따로 두지 않고, 사이드의 삽입 단계에서 경계 에지를 고려한다면 이렇게 형상에 따라서 수행속도가 크게 달라지는 경우를 제거할 수 있을 것이다.

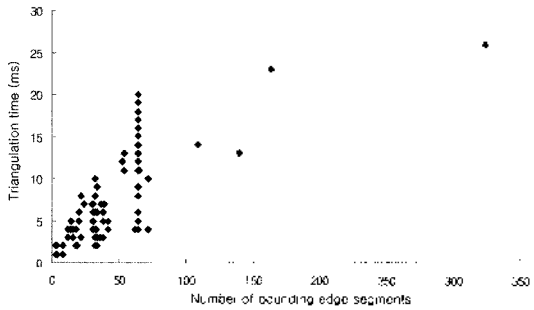

Fig. 13. Triangulation time.

5. 결론 및 향후 연구

일반적으로 삼각화 알고리즘이 데스크탑 환경에서도 컴퓨터 자원이 많이 소요되는 계산이기 때문에 현재까지 모바일 장치에서 직접 구현된 사례를 찾아보기 힘들다. 현재까지 모바일 장치에서의 3D 응용은 계산 성능이 월등한 데스크탑 환경에서 모델링되고 삼각화된 데이터 또는 렌더링된 이미지를 모바일 환경에서 효율적으로 보여주는데 초점이 맞추어져 있었지만 본 논문에서는 모바일 장치에서 CAD 데이터를 직접 가시화 하는 구현의 예를 보였다.

본 논문에서 설명한 *kd*-트리를 이용한 점진적 보로노이 다이어그램계산 알고리즘은 기존 알고리즘 대비 뛰어난 성능을 가진 것은 아니지만, 간단한 구현이 가능하여 모바일 장치와 같은 제한적인 환경에 쉽게 이식되고 실용 가능한 수행 속도를 보임을 확인하였다.

본 논문에서 설명된 알고리즘을 향상시키기 위하여 *kd*-트리와 버킷(bucket)[21] 방법을 병행하거나 그 밖의 가속 방법을 적용하는 것을 고려해 볼 수 있다. 또한 구현된 알고리즘은 현재 형상 계산의 안정성 때문에 64 bit 배정밀도 실수형을 사용하고 있는데, 단정밀도 실수 기반의 안정적 알고리즘[20]을 적용하게 되면 속도와 안정성의 두 가지 면에서 향상될 여지가 있다.

또한 설명된 알고리즘에서는 *kd*-트리를 이용하여 검색한 최단 거리 점으로부터 보로노이 영역을 구성하는데, 하프 에지 구조를 사용하지 않고 Delaunay 삼각화를 바로 수행한다면 보다 적은 메모리를 사용하여 삼각화를 구현할 수 있을 것이며 이를 위해서 최단거리 점으로부터 점진적 구성에 의해 수정되어야 할 삼각형들을 찾는 알고리즘의 개발이 필요하며 이를 통하여 보다 적은 메모리를 사용하면서 효율적인 삼각화 방법을 구현할 수 있을 것이다.

감사의 글

본 연구는 2006 중앙대학교 교내학술연구비 지원에 의해서 수행되었으며 이에 감사를 드립니다.

참고문헌

1. Woodward, C., Valli, S., Honkamaa, P. and Hakkarainen, M., "Wireless 3D CAD Viewing on a PDA Device", *Proceedings of the 2nd Asian International Mobile Computing Conference*, 14-17 May, 2002, Longkawi, Malaysia.
2. Su, X., Prabhu, B. S., Chu, C. C. and Gadh, R., "Middleware for Multimedia Mobile Collaborative System", *Proceedings of 3rd Annual Wireless Telecommunications Symposium*, May 14-15, 2004, CalPoly Pomona, Pomona, California, USA.
3. Khronos Group, "OpenGL ES Overview", http://www.khronos.org/opengles/
4. Direct3D Mobile for Windows Mobile-based Devices, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/mobilesdk5/html/mob5oriDirect3DMobile.asp
5. Mesh Generation : Software, http://www-users.informatik. rwth-aachen.de/~roberts/software.html
5. Zienkiewicz, O. C. and Phillips, D. V., "An Automatic Mesh Generation Scheme for Plane and Curved Surfaces by Isoparametric Coordinates", *International Journal for Numerical Methods in Engineering*, Vol. 3, pp. 519-528, 1971.
6. Ghassemi, F., "Automatic Mesh Generation Scheme for a Two or Three Dimensional Triangular Curved Surface", *Computers Struct*, Vol. 15, pp. 613-626, 1982.
7. Anastasiou, K. and Chan, C. T., "Automatic Triangular Mesh Generation Scheme for Curved Surfaces", *Communications in Numerical Methods in Engineering*, John Wiley & Sons, Ltd., Vol. 12, pp. 197-208, 1996.
9. Zunino, C., Lamberti, F. and Sanna, A., "A 3D Multiresolution Rendering Engine for PDA Devices", *SCI 2003 Proceedings*, Vol. 5, pp. 538-542.
10. Cavendish, J. C., Field, D. A. and Frey, W. H., "An Approach to Automatic Three Dimensional Finite Element Mesh Generation", *International Journal for Numerical Methods in Engineering*, John Wiley and Sons, Ltd., No. 21, pp. 329-347, 1985.
11. Lo, S. H., "Mesh Generation Over Curved Surfaces", *Asian-Pacific Conference on Computational Mechanics*, pp. 2345-50, Seoul, Korea 1996.
12. Fortune, S., "A Sweeping Algorithm for Voronoi Diagrams", *Algorithmica 2*, Springer-Verlag Inc., pp. 153-174, 1987.
13. Green, P. and Sibson, R., "Computing Dirichlet Tessellation in the Plane", *The Computer Journal*, Vol. 21, pp. 168-173, 1977.
14. Field, D. A., "Implementing Watson's Algorithm in Three Dimensions", *Proceedings of the 2nd Annu. ACM Symposium of Computer Geometry*, pp. 246-259, 1986.
15. O'Rourke, J., *Computational Geometry in C 2nd Ed.*, Cambridge University Press, pp. 162-163, 1998.
16. Kd-tree: Wikipedia, http://en.wikipedia.org/wiki/Kd-tree
17. Bentley, J. L., "Multidimensional Binary Search Trees Used for Associative Searching", *Communications of ACM* Vol. 18, No. 9, pp. 509-517, 1975.
18. Musser, D. R., Derge, G. J. and Saini, A., *STL Tutorial and Reference Guide. 2nd Edition*, Addson Wesley, 2001.
19. Hybrid Graphics Hompage, http://www.hybrid.fi/
20. Sugihara, K., Iri, M., Inagaki, H. and Imai, T., "Topology-oriented Implementation-An Approach to Robust Geometric Algorithms", *Algorithmica*, Vol. 27, No. 1, pp. 5-20, May, 2000.
21. Ohya, T., Iri, M. and Murota, K., "A Fast Voronoi-diagram Algorithm with Quaternary Tree Bucketing", *Information Processing Letters*, Vol. 8, Issue 5, pp. 227-231, May, 1984.

양 상 욱

1998년 중앙대학교 기계설계학과 학사
2000년 중앙대학교 기계설계학과 석사
2000년~2002년 삼성SDS UniCAD 개발팀 대리
2003년~2004년 eDiag Solutions Inc. 차장
2005년~현재 중앙대학교 기계공학부 박사과정

관심분야: 웹 기반 협력설계, Computational geometry, Mobile CAD, STEP 응용기술

최 영

1979년 서울대학교 기계설계학과 학사
1981년 KAIST 생산공학과 석사
1989년 Carnegie Mellon University 박사
1990년~1991년 KIST CAD/CAM 연구실 선임연구원
1992년~현재 중앙대학교 기계공학부 교수

관심분야: 형상모델링, Physically based modeling, 웹 기반 협력설계, STEP 응용기술